Deep Learning in Segregated Fund Valuation

Scarlet Ma, Di Ai, Siyuan Sheng, Ken Zhang

Abstraction

Segregated Fund is a special investment fund to provide capital appreciation with embedded insurance features. The traditional methodology to estimate the capital reserve and pricing of contracts goes to Monte-Carlo based stochastic models due to its complexity.

Recent research has introduced a deep learning model, Long Short-Term Memory (LSTM), to help cash flow projection for a Segregated Fund in its whole lifetime horizon. In this paper three new deep learning models are presented: Long Short-Term Memory with attention (LSTM-ATTN) to estimate the liability reserve and pricing Segregated Fund contracts, Wasserstein Generative Adversarial Network (WGAN) for stock return forecasting and Temporal Convolutional Network on GAN (TCN-GAN) for interest rate time series generation. As an example, the cash flow projection and Economic Capital for Segregated Fund portfolio are used to compare deep learning models against traditional models in terms of accuracy and computation efficiency.

Keyword: LSTM, attention mechanism, CNN, GAN, Deep Learning, Segregated Fund, Economic Capital

1. Introduction of Segregated Fund

Segregated Fund (Seg Fund) is a type of investment fund administered by Canadian insurance companies. In the form of deferred variable annuity contracts, it offers not only investment capital appreciation like mutual funds, but also predetermined guarantees to the policyholder, including level reimbursement of capital at the maturity of the contract (GMMB) and upon death of the policyholder (GMDB). The pool of investment of Seg Funds consists of securities such as bonds, stocks and equity indexes purchased by the insurance company. In nature, Seg Funds can be referred to as mutual funds with insurance features.

As the value of Seg Funds fluctuates with the market value of the underlying securities, the insurance company can be adversely affected if the account value drops below the guaranteed levels when their liability increases as the guarantees are in-the-money and payoffs are needed. Economic capital is held to mitigate against the market risk, while the projection of capital requirements requires complicated simulation and calculation works.

Other business features further add complexity and non-linearity to this process, such as the resetting feature of maturity guarantees that increases the guaranteed level to match higher account values.

Current Methodology in Calculating Capital Requirement

Nested Monte-Carlo based stochastic models have been used in the calculation of capital requirements, reserving and dynamic hedging for Seg Fund products by insurers who are allowed to use "internal models" under Solvency II.

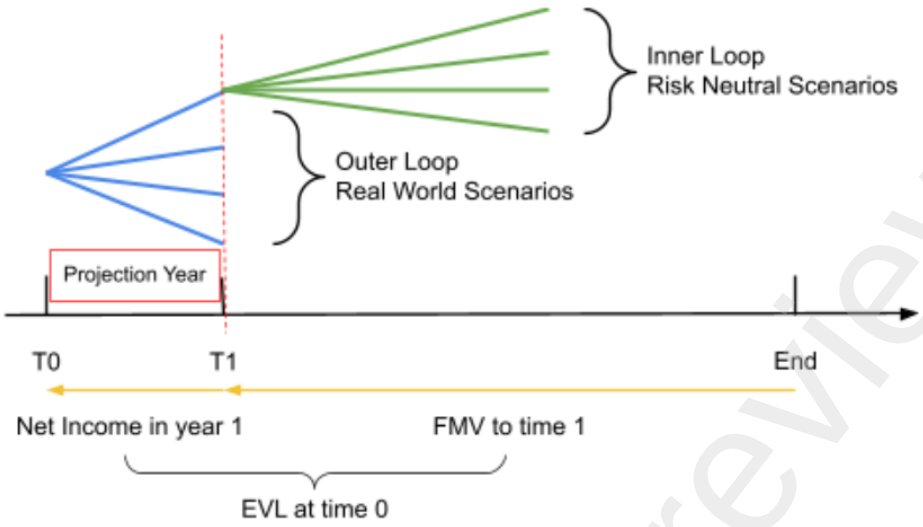
The core to these models is the nested stochastic-on-stochastic structure, formed by several "inner" loop scenarios diverging from each of the "outer" loop scenarios at a future time point. The scenarios are simulated by stochastic methods based on underlying economic risk variables from real-world or risk-neutral probability distribution. This two-layer loop of scenarios are then passed into a liability cash flow system (such as AXIS)[1] to project cash flows over the time horizon of the product and then to estimate the desired statistics such as best estimate or Value-at-Risk (VaR).

Figure 1 Nested Monte-Carlo Model Structure for Seg Fund Cash Flow Projection

[1] GGY AXIS a widely used actuarial software in the insurance industry

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847



This nested structure has been applied to the valuation of Seg Funds. The economic value of liability (EVL) of the underlying portfolio is defined as the present value of net cash flows including incomes from fund investment and premiums net of outcomes for expected guarantees and expenses. It is projected as present value at time 0 of the average fair market value of the liability (FMV) across a set of risk neutral "inner" scenarios discounted to the end of year 1 plus the projected net income during the first year from the real world "outer" scenario:

$$EVL = PV(FMV(t_1) + Y_1\ net\ income)$$

Further incorporating any hedge credit, the net EVL can be then used for capital requirement calculation, pricing or reserving purposes of Seg Funds.

For example, Economic Capital can be then calculated as the difference between the 99.95% value-at-risk and the best estimate of projected EVL distribution.

$$EC = VaR_{99.95}(EVL) - E(EVL)$$

The complexity of guaranteed benefit features associated with Seg Fund calculation-intensive tasks brings considerable computational challenges to the nested loop structure of the full stochastic scenario generation process. Due to the computation constraints of traditional actuarial systems (such as Moody's GGY AXIS) running the full process on a typical Seg Fund portfolio can consume a large amount of computational workforce and days of processing time, mostly from the valuation process on inner loop scenarios. AXIS also has the limitation that the same set of scenarios should be used for all seriatim policies in the portfolio, regardless of the specific underlying fund composition or product features.

The Least-Square Monte-Carlo method (LSMC)[2] can be used to significantly reduce the required number of total scenarios through averaging across all inner loop scenarios to achieve an acceptable degree of accuracy as compared with projections from the full approach, and thus improves the efficiency of the valuation process. The valuation of Seg Funds shows a high similarity to that of the American options. Inspired by the LSMC method, a scenario selection model using linear regression on key variables has been proposed to reduce the processing time of the nested stochastic process.

[2] proposed by Longstaff and Schwarz is an alternative approach to valuation of American variable options, which leverages the analytical approximation (for example, linear regression), of the liability

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

2. Deep Learning Methodology

Recent research [1] demonstrated that a deep learning algorithm, Long Short-term Memory (LSTM), can be trained to project path-dependent cash flows at multiple timesteps of insurance portfolios under various types and combinations of stochastic scenarios. Instead of estimating specific statistics (such as expectation or VaR), the author developed an alternative cash flow model as a multi-purpose solution to various calculation tasks from reserve valuation, risk-based capital calculation and pricing. The nature of the insurance cash flow model shares many similarities with a time series problem as a rule to derive the output time series of cash flows by input time series of multiple risk factors as defined in scenarios. Cash flows predicted at a certain time horizon should also impact subsequent cash flows thereafter. Patterns in the early time range of the scenarios should also continuously affect the predictions for future time. Instead of explicitly expressing all these relationships, the author proved that the LSTM network can simultaneously learn the complexity by selecting information from input scenarios to store in memory for prediction, as demonstrated through two examples of cash flow prediction under different types of scenarios. Test results in the paper showed that the LSTM-based model can achieve a high degree of accuracy in reproducing cash flows with much less computation cost than the traditional Monte-Carlo simulation and LSMC models.

In this paper we illustrate in detail how the LSTM algorithm is fitted to capture the non-linear relationship between the output time series of cash flows and the input time series of stochastic economic scenarios. Suitable tuning techniques are applied to tune the projection accuracy in the PyTorch machine learning library.

We also advance to integrating the Attention mechanism to our LSTM network, considering its proven effectiveness in solving the long-time dependency problem of financial time series predictions in recent research [4], and compare the prediction accuracy and model performance of the deep learning models against the original benchmark LSMC model. Finally, we explore the applications of Generative Adversarial Network (GAN) to the scenario generation of equity returns and interest rates.

In the following sections of this paper, we detail our applications of Deep Learning models to an example block of business composed of Seg Fund contracts with a mixture of maturity (GMMB) and death (GMDB) guaranteed benefits levels and realistically diversified policyholder characteristics (attained age, gender, anniversary etc.), and compare their results with the benchmark cash flow model applying the LSMC method. Through two experiments, we predict the Seg Fund cash flows for 12 months and 25 years, using the economic value of net future liabilities as the target output, and the following six risk factors as inputs:

- Yield Curve of 3-month, 6-month, 12-month and 2-year interest rates: scenarios generated by Cox-Ingersoll-Ross (CIR) short rate model
- Monthly equity index volatility: scenarios produced by Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) model
- Monthly equity index return: assuming that the stocks follow geometric Brownian motion, a continuous-time stochastic process, calculated as:

$$Index\ Return_t = e^{\left(\mu - \frac{\sigma^2}{2}\right)t}$$

The combination of 3 risk factors is used as input to our models. The data details refer to section 3.

LSTM Specialized RNN

Traditional models for Seg Fund cash flow calculation are developed in the widely used GGY AXIS platform. At each time step across the policy life, the in force fund value (V) from the previous step, plus economic risk factors (X) and other static actuarial assumptions from the current step are fed into complicated calculations empowered by AXIS, to project the cash flows, to update policy values after complex policy features such as resetting of guaranteed benefits (G), and to output the fund value from policies in force at the current time point. Mathematically, the relation among risk factors, fund value and our target cash flow can be formulated as follows.

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

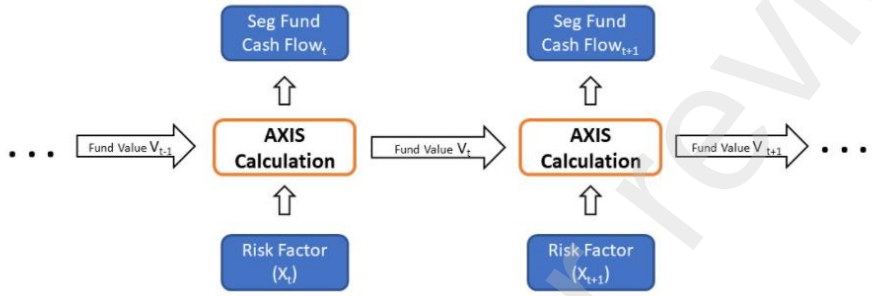
Denote f(•) and g(•) as the embedded computational relationships modelled in AXIS, and V_t, X_t and G_t as the fund value, risk factors and effective guaranteed benefits after any applicable resetting at time t, respectively:

$$V_t = f(V_{t-1}, X_t)$$

Where V_0 is determined based on the initial deposit from policyholder

$$Seg\ Fund\ Cash\ Flow_t = g(V_t, G_t)$$

Figure 2 Cash Flow Projection by GGY AXIS



Such calculation in GGY AXIS is computationally intensive and time consuming, therefore in the industry, some estimation methods proposed to shorten and simplify the process by utilizing estimation models such as multivariate regressions. One limitation of the regression model or any parametric model is that they assume a certain linear or non-linear relationship between risk factors and cash flow outputs. However, complicated product features, such as the guarantee resetting feature, of Seg Fund policies counter such assumptions as the fund values and cash flows are path dependent.

In this paper, we propose a solution to the path-dependent cash flow projection problem by fitting a Long-term short memory (LSTM) recurrent neural network (RNN), to approximate the unknown function f(•) and g(•).

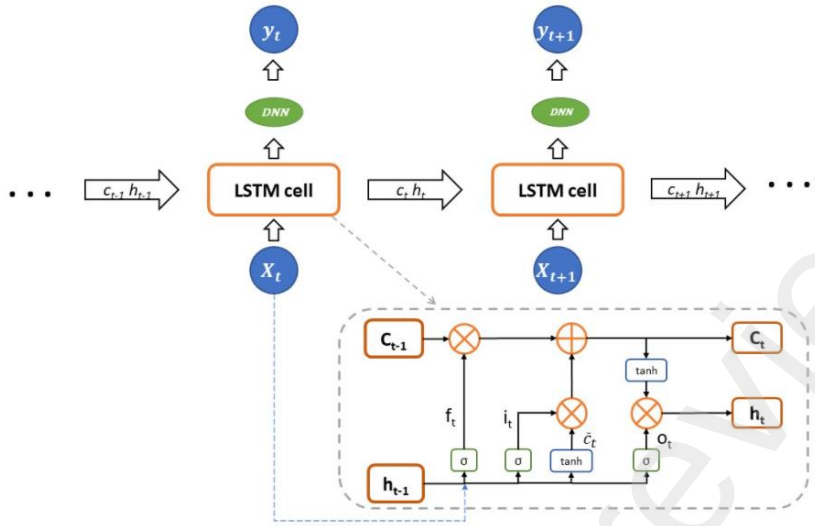
RNN is widely used in modelling sequential data like language translation and voice recognition. The architecture of RNN makes it also applicable to time series data. LSTM was introduced as a special type of RNN and has shown proven capability of learning long-term dependency through the gating mechanisms in sophistically structured cells. We consider applying the LSTM model to the projection of path dependent Seg Fund cash flows.

LSTMs have chain-like structures like RNNs where the previous hidden layer and current input are fed into a cell to produce the next output recursively, with a different design of repeating cells across the time sequence.

Figure 3 Illustration of the LSTM Model Architecture

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847



A LSTM cell is composed of an input gate, an output gate and a forget gate, controlling the income and output flow of information in the cell. Mathematically, denote k as the hyperparameter of the dimension of states:

$$Forget\ gate: f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \epsilon\ (0,1)^k$$
$$Input\ gate: i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \epsilon\ (0,1)^k$$
$$Cell\ input: i_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \epsilon\ (-1,1)^k$$
$$Output\ gate: o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \epsilon\ (0,1)^k$$
$$Cell\ state: c_t = f_t \cdot c_{t-1} + i_t \cdot c_t \epsilon\ R^k$$
$$Hidden\ state: h_t = o_t \cdot \tanh(c_t) \epsilon\ R^k$$

Where the activation functions are defined as:

Activation Function		Definition
σ_g	Sigmoid	$\frac{1}{1+e^{-x}}$
$tanh$	Hyperbolic tangent function	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$
$ReLU$	Rectified Linear Unit	$\max(0, x)$

Over the time sequence, the input layer consists of risk factor scenarios, formed as a matrix:

$$X = [x_1, x_2, \ldots x_n] \in\ R^{n \times m}$$

where n is the length of time window, m is the number of features.

The hidden layer stores information from all hidden states across the sequence:

$$H = [h_1, h_2, \ldots h_n] \in\ (-1,1)^{n \times k}$$

Following the LSTM layers, at time step level, we further incorporated a feed-forward deep neural network (FFDNN) component, consisting of two fully connected layers with ReLU activation functions plus one linear layer to produce the target predictions.

$$Prediction: y_t = FFDNN(h_t)$$

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

Specifically, FFDNN is composed of a set of layers of neurons. Each neuron of the network is connected to other neurons through a numeric transformation called weight. Additionally, each neuron has an activation function through which it transforms its own global input into output. With predefined dimension hyperparameters l_1 and l_2:

$$L1: h_t^1 = ReLU(W_1 h_t + b_1), where\ W_1 \in R^{l_1 \times k}, b_1 \in R^{l_1}$$
$$L2: h_t^2 = ReLU(W_2 h_t + b_2), where\ W_2 \in R^{l_2 \times l_1}, b_2 \in R^{l_2}$$
$$L3: y_t = W_3 h_t^2 + b_3, where\ W_3 \in R^{1 \times l_2}, b_3 \in R^1$$

The output layer is the predictions from all time steps:

$$Y = [y_1, y_2, \dots y_n] \in R^{n \times 1}$$

$W_* \in R^{m \times k}, U_* \in R^{k \times k}, b_* \in R^k$ for $* \in \{f, i, c, o\}$, and $W_\#, b_\# \ for\ \# \in \{1,2,3\}$ are parameters that are needed to be learned during the training process.

In our example, x_t consists of three risk factor - 3-month interest rate, equity return and equity volatility at step t, and y_t is the net cash flow. LSTM model learns the relation between x_t and y_t through backpropagation, achieving the same computational functionality with GGY AXIS of cash flow projection. One inherent assumption from the LSTM model is that the learnable parameters are independent of time t across the sequence in a single training scenario. In real business, the underlying relationship between economic risk factors and output cash flows can change from time to time, for example, affected by time-dependent actuarial inputs. However, for an entire portfolio of well-diversified policies, the mathematical relationships can be assumed to remain effectively unchanged across the policy life without expressing explicitly.

LSTM with Attention (LSTM-Attn)

A primary constraint of the encoder-decoder based RNN / LSTM networks has been observed that their constitutionally sequential nature limits the model performance in long-range sequence modelling, in that the output at current timestep should represent information from all previous inputs in the sequence as represented by hidden states, resulting in deterioration of the model's prediction power and performance due to memory constraints at longer time series.

Attention mechanisms have emerged in deep learning to seek for an improvement for long-range sequence modelling. It trains the model to assign different attention weights across the whole input feature sequence and then to selectively align the output with the optimised input sequence. In this way, attention-integrated models have shown proven capability to effectively avoid the long-dependency problem, as well as enhanced model interpretability such as feature importance.

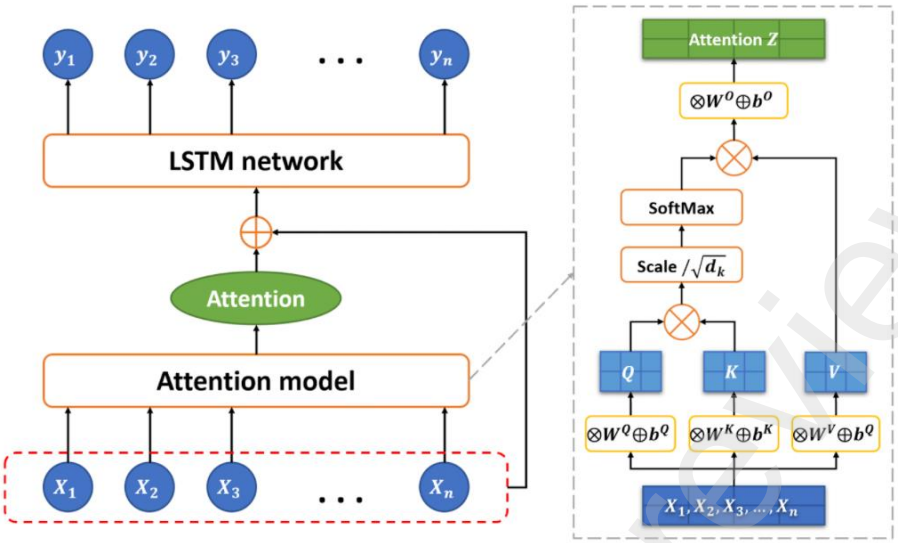
Since its first introduction [5], the attention integrated neural networks have been widely applied to complex sequence learning problems such as image recognition, machine translation and natural language processing. Recent research [4] has also proved the application to financial time series prediction problems of the LSTM network with the integral of attention mechanism. In this paper, we consider implementing the LSTM with attention model (LSTM-Attn) to enhance the cash flow model's performance potential and interpretability.

The architecture of our LSTM-Attn model consists of two components: the attention model and the LSTM model.

Figure 4: Illustration of the LSTM-Attn Model Architecture

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847



To develop the attention model, we apply the Scaled Dot-Product Attention proposed by [3]. Specifically, using the query, the key and the value as input, the output Attention is calculated as the weighted sum of the values, where the weight assigned to each value is computed by the query and the corresponding key.

The queries, keys and values are defined as matrices of dimension md, where m is the number of input features and d is a dimension parameter. The matrices are produced by linear feed forward layers on the input feature sequence, which apply trainable weight matrices W^Q, W^K and W^V and additive trainable biases b^Q, b^K and, b^V:

$$X \cdot W^Q + b^Q = Q \in R^{m \times d}$$
$$X \cdot W^K + b^K = K \in R^{m \times d}$$
$$X \cdot W^V + b^V = V \in R^{m \times d}$$

Then the dot products are computed of all the queries and keys, each scaled by the square-root of d, and a SoftMax function is applied to the scaled products to calculate the attention scores. Lastly, the dot product of the scores and the values is computed and processed through another linear feed forward layer of weight W^0 and bias b^0 to generate the final Attention output Z.

$$Z_0(Q,K,V) = SoftMax\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V$$
$$Z = Z_0 \cdot W^0 + b^0$$

The sum of the Attention output and the original input sequence is subsequently fed as the newly optimized input sequence into the standard LSTM network, followed by a FFDNN component of three fully connected feed-forward layers plus one linear layer to predict our target values, i.e., the cash flows.

$$L1: h_t^1 = ReLU(W_1 h_t + b_1), where\ W_1 \in R^{l_1 \times k}, b_1 \in R^{l_1}$$
$$L2: h_t^2 = ReLU(W_2 h_t^1 + b_2), where\ W_2 \in R^{l_2 \times l_1}, b_2 \in R^{l_2}$$
$$L3: h_t^3 = ReLU(W_3 h_t^2 + b_3), where\ W_3 \in R^{l_3 \times l_2}, b_3 \in R^{l_3}$$
$$L4: y_t = W_4 h_t^3 + b_4, where\ W_4 \in R^{1 \times l_3}, b_4 \in R^1$$

Generative Adversarial Network (GAN)

In the previous sections we proposed LSTM and LSTM-Attn to predict the cash flow. Another methodology, Generative Adversarial Network (GAN), has emerged, with promising applications in generative modeling tasks

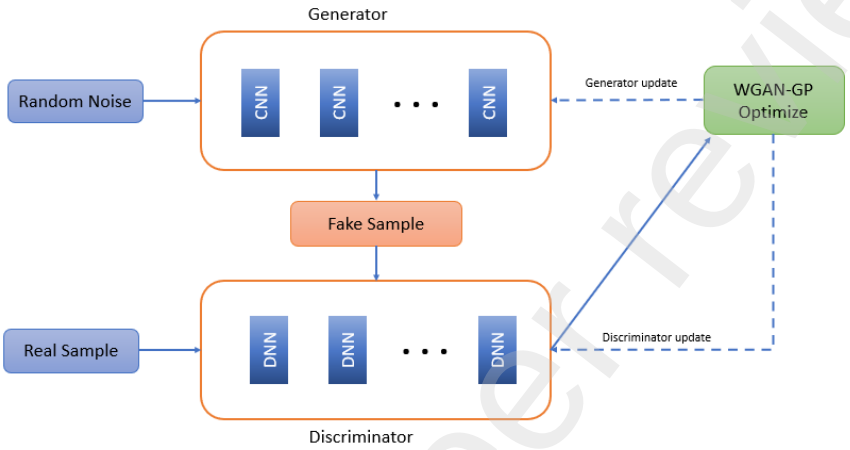
Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

such as object detection, image translation and generations. Recent research has proposed using unsupervised learning techniques such as Wasserstein GAN [6] and Temporal Convolutional Network on GAN for generating synthetic values that follow the probability density of the observed values.

On top of the WGAN, improved training of Wasserstein GANs was proposed by Ishaan Gulrajani [7] to address the convergence problem in WGAN. In this paper, we experiment with the application of WGAN-GP to the simulation of future stock returns as an alternative to complement traditional scenario generation methods such as GARCH. The process is demonstrated as below:

Figure 5 WGAN-GP process



The generator takes random noise and produces synthetic returns based on historical returns. The discriminator takes the return time series from the generator as input and produces the score of the time series. With its generator transforms random number to real return, and its discriminator is trained to maximize:

$$L(D, g_\theta) = E_{x\sim P_r}[logD(x)] + E_{x\sim P_\theta}[\log(1 - D(x))]$$

By investigating the application of WGAN in the Seg Fund case, we attempt to find a complementary method to improve the equity return generation by the GARCH model, producing relatively reasonable simulations in terms of return scale over a long-time horizon. Intuitively, the WGAN model works by training the generator to learn to transform randomly generated noises from Gaussian distribution into a synthetic stock return distribution that eventually fools the discriminator, that is, to produce returns that follow the same patterns with the actual historical returns.

We also experiment with the application of TCN-GAN as an alternative to the traditional CIR methodology for generating stochastic interest rate scenarios.

TCN is a variation of Convolutional Neural Network, which has recently been shown to be competitive on many sequence modeling tasks. TCNs employ dilated convolutions to capture time series with a long horizon, which insert holes by skipping consecutive elements. The details of the TCN framework can be found in [8].

In our TCN-GAN model, we employed TCN as the generator. During the training process, the generator is trained using samples of the historical data of the specified size. Random noise that follows a multivariate standard normal distribution is generated. The estimator of the sample, a time series of the length of the training horizon, is then estimated by inferring the random noise through the TCN generator. The discriminator is represented by a TCN with the output being a binary variable of [0,1] to distinguish the estimated time series from the drawn sample data. We train the generator and discriminator iteratively. The whole process of training GAN is summarized in the Algorithm below.

Denote G the generator, Denote D the discriminator, and M the training sample size. The process of TCN-GAN is described as follows.

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

1) Set the learning rates of generator and discriminator as p_G and p_D.
2) Initialize weight W_G and W_D.
3) Repeat the following training iterations until convergence:

Get the training sample from historical input data, $X_M = \{x_1, x_2 \ldots x_M\} \epsilon R^M$
Generate random noises $N_M = \{n_1, n_2 \ldots n_M\}$,
Generate the estimator of X_M; $\widetilde{X_M} = G(N_M)$
Compute the loss L_D and store the gradient Δ_D that minimize the loss function:

$$L_D = \frac{1}{M}\sum_{i=1}^{M} log(D(\widetilde{X_M})) + log(1 - D(X_M))$$

Update the discriminator's parameters: $W_D = W_D + p_D \cdot \Delta_D$
For every k step, do:

Compute the loss L_G and store the gradient Δ_G that minimize the loss function:
$L_G = \frac{1}{M}\sum_{i=1}^{M} log(D(G(\widetilde{X_M})))$
Update the generator's parameters: $W_G = W_G + p_G \cdot \Delta_G$

3. Training and Testing

Data and Evaluation Metric

Before applications of neural network models, we first need to generate the training dataset (X_t, y_t). In the first exercise, we used a 12-month window time series where t =1,2 …12. Each risk factor in X_t is generated independently using Monte-Carlo simulation. In the second example, an 8-year time series where t = 1,2 …8 is used. Once the input scenarios are created, they can be fed into GGY AXIS to produce the output value, y_t. Market data are utilized to calibrate for the CIR and GARCH stochastic models for scenario generation and to train the GAN models, as follows:

1) Canadian Dollar Offered Rate (CDOR), used for CIR calibration and the training of TCN-GAN to produce forward interest rates (1994 to 2021), adjusted by centering at zero.
2) Historical monthly TSX/S&P Index close prices, used for GARCH calibration (2001 to 2021) and the training of WGAN-GP to produce future index returns (1985 to 2021).

The deep learning algorithms are trained with 40,000 scenarios of economic risk factors, and 10,000 scenarios separately simulated for validation. To speed up the training process, the training dataset scenarios are grouped into mini batches of size 5.

We compare the performance of different cash flow simulation methods (traditional LSMC, LSTM and LSTM-Attn) by two evaluation metrics, mean squared error (MSE) which measures the average squared difference between the estimated and the actual values, and mean absolute error (MAE), which measures the average absolute errors between the estimated and the actual values. For estimates $\hat{Y}_i = \{Y_{i,t}\}$and actual values $Y_i = \{Y_{i,t}\}$ t=0,,n i=1,2,...,N samples, they are defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2 = \frac{1}{N*n}\sum_{i=1}^{N}\sum_{t=1}^{n}\left(Y_{i,t} - \hat{Y}_{i,t}\right)^2$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i| = \frac{1}{N*n}\sum_{i=1}^{N}\sum_{t=1}^{n}|Y_{i,t} - \hat{Y}_{i,t}|$$

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

The synthetic returns generated from TCN-GAN and WGAN models are evaluated by distribution and path comparison against the actual historical returns, and the simulated rates from CIR and GARCH models.

Training and Testing

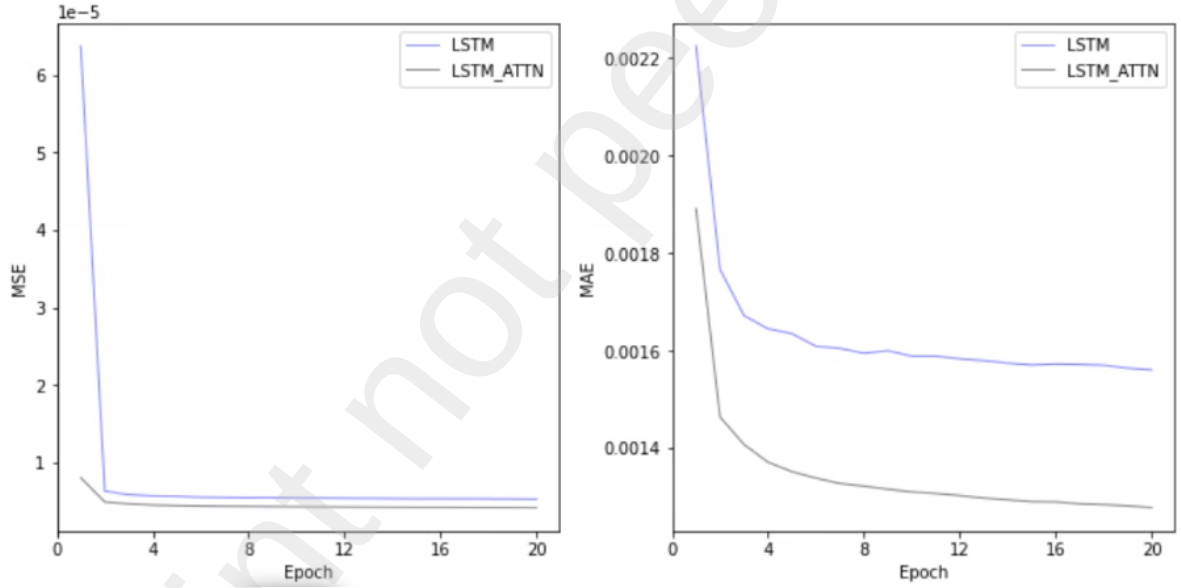
The LSTM model has two major hyperparameters: the number of LSTM layers, and the size of hidden states k, and the FFDNN has three layers with dimensions of l_1, l_2 and 1 respectively. After repeating the training process on several combinations.

To speed up the training process, we choose a batch size = 5. The final hyperparameter settings are: 1 layer of LSTM with $k = 300$, $l_1 = 500$ and $l_2 = 100$.

In addition to parameters used in the LSTM component, the attention mechanism LSTM-Attn has an additional hyperparameter setting for the dimension of query, key, and value matrices. From our experiments, we use 1 layer of LSTM with $k = 400$, and for the subsequent FFDNN component of 4 layers with $l_1 = 500$, $l_2 = 400$ and $l_3 = 100$.

The training process of the LSTM and LSTM-Attn models demonstrates that, after a sudden drop through the first two epochs, the MSE and MAE performance metrics for both models continuously declined to statistically low levels by 20 epochs, while the LSTM-Attn model consistently outperformed the LSTM model across the epochs. After 20 epochs, the LSTM and LSTM-Attn model results converged sufficiently.

Figure 6 Training Process of LSTM & LSTM-Attn by Epoch



Main hyperparameters used in WGAN-GP include the conditional number, which is the number of previous returns used to generate future returns, the time series dimension, which is the total time steps of time series passed into the generator, and the latent dimension controlling the dimension of generated random noises. In our case, we choose a condition number of 5, time series dimension of 17, and latent dimension of 10, and this implies that we use previous 5 months return to generator future 12 months return (in total 17 months), and the returns are transformed by the generator from random noises of dimension 10.

The core hyperparameter used in TCN-GAN specifies the training sample size M, controlling the length of the time series. In our experiment, we chose the sample size of 120 days to train the model. Once the model is trained, the final outputs can be of any dimension, or in other words, the generated time series of interest rates can be of any length.

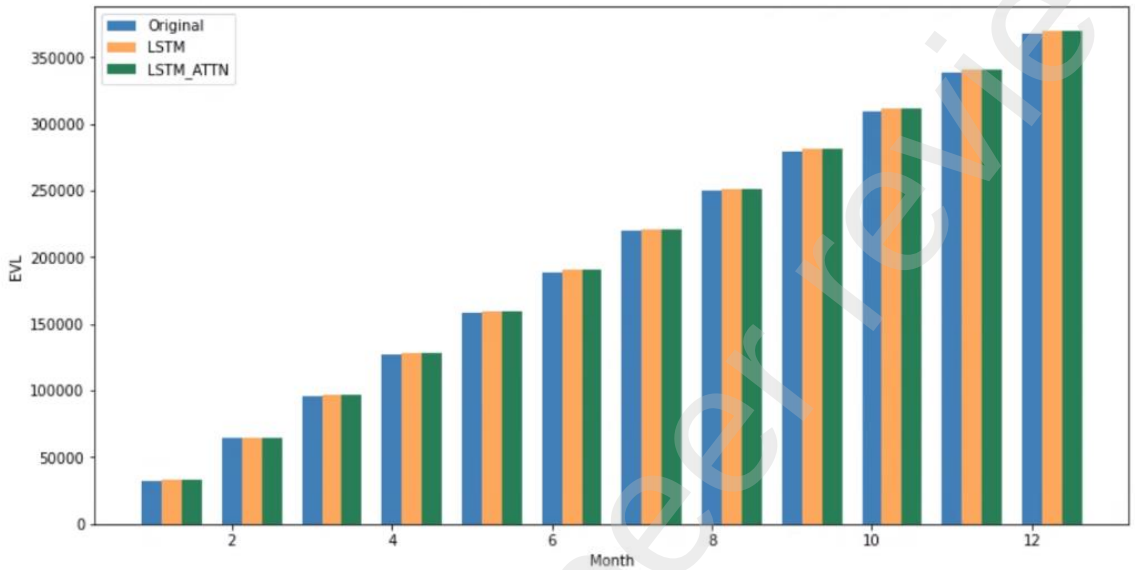
Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847

Results and Comparisons

Figure 7 illustrates the result comparison of our deep learning models against the base method (LSMC) and demonstrates that the DL methods have achieved satisfactory degrees of accuracy with errors around 0.1%.

Figure 7: Projected Liability Cash Flow Comparison



Additionally, we calculate the one-year economic capital requirements as the difference between VaR (99.95) and the average of projected cash flows (in present value discounted with a discounting rate of 1.79%), and find the results of the LSTM and LSTM-Attn models to be lower than but still relatively close to the original LSMC results, with LSTM-Attn producing more precise tail projections, as shown in Table 1.

Table 1 Comparison of Economic Capital Results, One-Year Projection Period

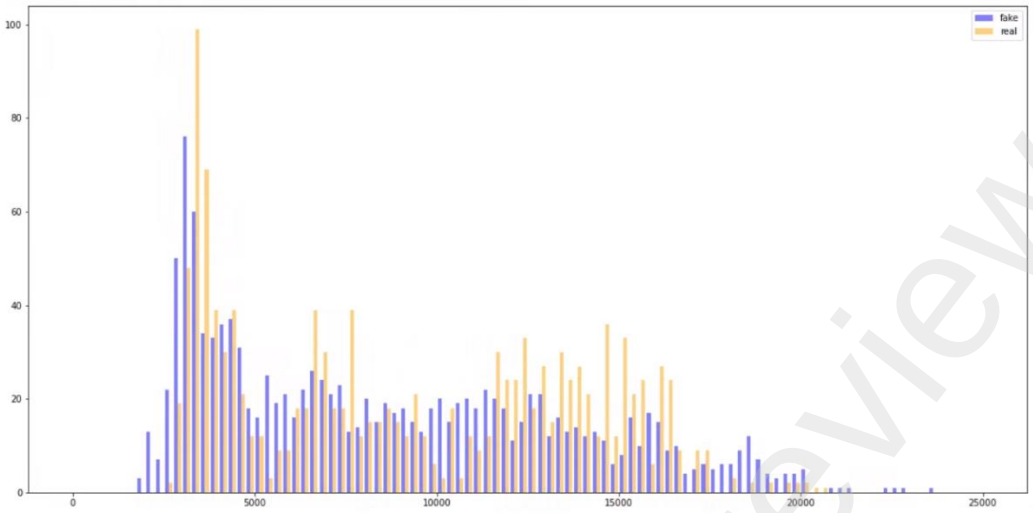
	Original LSMC	LSTM	LSTM Attn
Economic Capital @ 1.79%	106,728.90	96,419.11	101,680.90

The comparison of simulations from WGAN ("fake" returns) against the historical distribution of the in-sample data ("real" returns) in terms of monthly return is illustrated by the histogram of Figure 7.

Figure 7: Comparison of WCAN-GP Simulations and Historical Returns

Preprint not peer reviewed

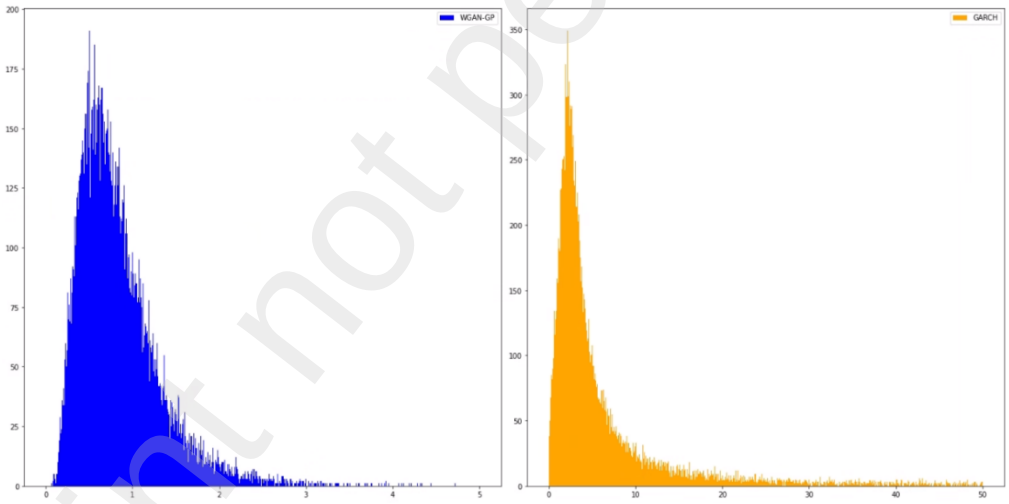
This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847



As seen in Figure 8, the back testing results using overlapped validation data have demonstrated that the simulated distribution by the generator has successfully learnt the pattern and correlation between past return and future returns.

We then examine the simulation of future returns by WGAN and compare the simulations with GARCH-BS against the test data in terms of 8-year accumulated return distribution, as shown by the following histograms of Figure 9.

Figure 9: Comparison of Simulated Future Returns, WGAN vs. GARCH-BS



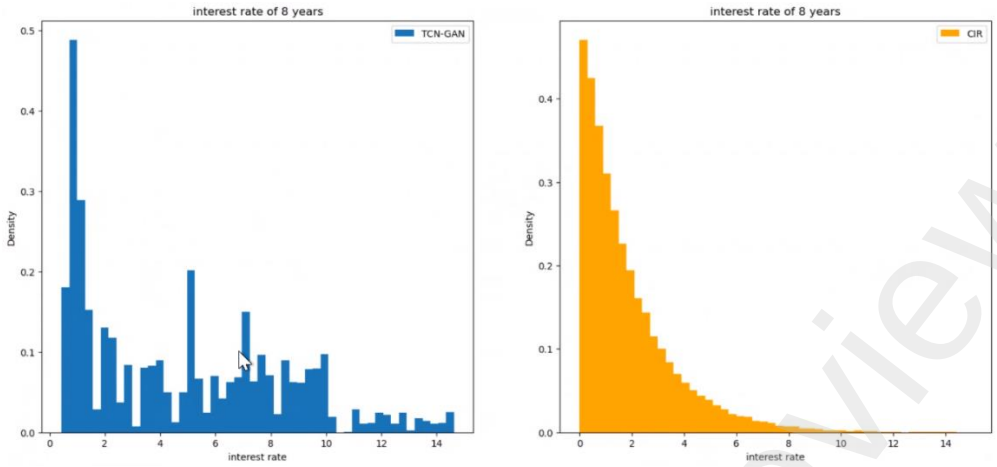
Overall, the distribution of the WGAN simulated returns exhibits similar patterns with the GARCH-BS simulations. However, it is notable that over a relatively long-time horizon, the GARCH model simulates index returns that may accumulate to unreasonably high levels, reaching a maximum 50 times of the initial rate in 8 years, further exploding as simulated into longer periods. Meanwhile, the WGAN model can produce accumulated index returns within a reasonable range, supporting it as an improvement to the traditional generators.

The simulation of interest rate generated by TCN_GAN is compared to those by CIR. Interest rates at the end of 8th year of both methods are shown in Figure 10.

Figure 10: Comparison of Simulated Interest Rate, TCN-GAN vs. CIR

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847



It is found that although both methods generated most values around zeros, the distribution of TCN-GAN contains more larger values than CIR. Overall, the distribution of TCN_GAN simulated interest rates exhibit similar patterns with the CIR simulations.

4. Conclusion

Complicated features of the Seg Fund cash flows have posed computational challenges to the traditional cash flow projection models, such as the nested Monte-Carlo stochastic models, for valuation, capital estimation and pricing purposes. In addition, the current stochastic simulation methods for interest rate and equity return generations, such as CIR and GARCH models, encountered difficulties for parameter calibrations, and exhibited limitations over long time horizons and for massive scenario generations.

In this paper, we demonstrated the applications of deep neural networks techniques based on LSTM and LSTM with Attention models to cash flow projections for Seg Funds, by interpreting the scenarios of interest rate and equity return time series into liability cash flows. Particularly, the LSTM architectures are found to be good solutions to the problem of Seg Fund liability cash flow projections, achieving satisfactory accuracy and enhanced performance efficiencies in terms of time and computation requirements. We also experimented with Generative Adversarial Networks for time series generation of interest rates and equity returns and found that the models succeeded in generating economic scenarios while bypassing the limitations in traditional GARCH and CIR models.

Reference

[1] Aubrey Clayton, Deep Learning the Cash Flow Model, Jun 3 2020

[2] Goodfellow et. al., Deep Learning, 2016

[3] Ashish Vaswani et. al., Attention Is All You Need, v5, arXiv:1706.03762v5, Dec 6 2017

[4] Xuan Zhang et. al., AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction, IOP Conference Series, IOP Conf. Ser.: Mater. Sci. Eng. 569 052037, 2019

[5] Dzmitry Bahdanau et. al., Neural Machine Translation by jointly learning to align and translate, v7, ICLR, arXiv:1409.0473v7, May 19, 2016

[6] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, Jan 26, 2017

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, Improved Training of Wasserstein GANs, Mar 31, 2017

[8] Magnus Wiese et. al., Quant GANs: Deep Generation of Financial Time Series, v2, 21 Dec 2019

Preprint not peer reviewed

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4074847